Depoy Rancher Server (single instance)

1. Create VPC with the name “FOLIO-Rancher-Server-VPC”
   a. Enable “DNS hostnames” for the VPC
2. Create Internet Gateway with the name “FOLIO-Rancher-Server-IGW”
   a. Attach the Internet Gateway to the“FOLIO-Rancher-Server-VPC”
3. Create a subnet with the name “FOLIO-Rancher-Server-subnet”
4. Create a route table with the name “FOLIO-Rancher-Server-public-RT”
   a. Add a route 0.0.0.0/0 -> “FOLIO-Rancher-Server-IGW”
   b. Associate “FOLIO-Rancher-Server-subnet” with the this “FOLIO-Rancher-Server-public-RT”
5. Create a subnet with the name “FOLIO-Rancher-Server-subnet-internal” in the same availability zone as “FOLIO-Rancher-Server-subnet”
6. Create a NAT Gateway with the name “FOLIO-Rancher-Server-NAT-GW” for “FOLIO-Rancher-Server-VPC” and “FOLIO-Rancher-Server-subnet”
7. Create a route table with the name “FOLIO-Rancher-Server-private-RT”
   a. Add a route 0.0.0.0/0 -> “FOLIO-Rancher-Server-NAT-GW”
   b. Associate “FOLIO-Rancher-Server-subnet-internal” with the this “FOLIO-Rancher-Server-private-RT”
8. Create a security group with the name “FOLIO-Rancher-Server-SG” linked to the “FOLIO-Rancher-Server-VPC”
   a. Add an inbound rule to the “FOLIO-Rancher-Server-SG” [ssh, tcp, 22, Anywhere] - ssh access
   b. Add an inbound rule to the “FOLIO-Rancher-Server-SG” [https, tcp, 443, Anywhere] - access to the server
   c. Add an inbound rule to the “FOLIO-Rancher-Server-SG” [https, tcp, 80, Anywhere] - health check access
9. Create a key pair with the name “FOLIO-Rancher-Server-Key-Pair”
10. Create a EC2 instance with the name “FOLIO-Rancher-Server”
    a. Use “FOLIO-Rancher-Server-VPC” for VPC
    b. Use “FOLIO-Rancher-Server-subnet-internal” for subnet
    c. No public IP
    d. Use “FOLIO-Rancher-Server-SG” security group
    e. Use “FOLIO-Rancher-Server-Key-Pair”
11. Create a target group with the name “FOLIO-Rancher-Server-pub-SSH-TG”
    a. Target type: instance
    b. Protocol: TCP
    c. Port: 22
    d. VPC: “FOLIO-Rancher-Server-VPC”
    e. Health check setting
    f. Protocol: TCP
12. On the “Targets” tab of the “FOLIO-Rancher-Server-pub-SSH-TG” add “FOLIO-Rancher-Server” on the port 22
13. Create a public network load balancer “FOLIO-Rancher-Server-SSH-nlb”
    a. Scheme: internet-facing
    b. Load Balancer Protocol: TCP
    c. Load Balancer Port: 22
    d. VPC: “FOLIO-Rancher-Server-VPC”
    e. Availability Zone: AZ with subnet “FOLIO-Rancher-Server-subnet”
    f. Next
    g. Next
    h. Target group: Existing Target Group
    i. Name: “FOLIO-Rancher-Server-pub-SSH-TG”

14. Open SSH terminal to the “FOLIO-Rancher-Server” using DNS name of the “FOLIO-Rancher-Server-SSH-nlb” and install docker using https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html#install_docker
15. Install Rancher Server using
    docker run -d --restart=unless-stopped --name rancher-server --ulimit nofile=4096:4096 -p 80:80 -p 443:443 rancher/rancher:latest
16. Create a target group with the name “FOLIO-Rancher-Server-internal-TG”
    a. Target type: instance
    b. Protocol: TCP
    c. Port: 443
    d. VPC: “FOLIO-Rancher-Server-VPC”
    e. Health check setting
    f. Protocol: HTTP
    g. Path: /healthz
    h. Advanced health check settings
    i. Port: overide - 80
17. On the “Targets” tab of the “FOLIO-Rancher-Server-internal-TG” add “FOLIO-Rancher-Server” on the port 443
18. Create a private network load balancer “FOLIO-Rancher-Server-nlb-int”
    a. Scheme: internal
    b. Load Balancer Protocol: TCP
    c. Load Balancer Port: 443
    d. VPC: “FOLIO-Rancher-Server-VPC”
    e. Availability Zone: AZ with subnet “FOLIO-Rancher-Server-subnet-internal”
    f. Next
    g. Next
    h. Target group: Existing Target Group
    i. Name: “FOLIO-Rancher-Server-internal-TG”
19. Create a target group with the name “FOLIO-Rancher-Server-pub-TG”
    a. Target type: instance
    b. Protocol: TCP
    c. Port: 443
    d. VPC: “FOLIO-Rancher-Server-VPC”
    e. Health check setting
    f. Protocol: HTTP
    g. Path: /healthz
    h. Advanced health check settings
    i. Port: overide - 80
20. On the “Targets” tab of the “FOLIO-Rancher-Server-pub-TG” add “FOLIO-Rancher-Server” on the port 443
21. Create a public network load balancer “FOLIO-Rancher-Server-nlb”
    a. Scheme: internet-facing
    b. Load Balancer Protocol: TCP
    c. Load Balancer Port: 443
    d. VPC: “FOLIO-Rancher-Server-VPC”
    e. Availability Zone: AZ with subnet “FOLIO-Rancher-Server-subnet”
    f. Next
    g. Next
    h. Target group: Existing Target Group
    i. Name: “FOLIO-Rancher-Server-pub-TG”

22. Open Rancher Web Admin Console in a browser using DNS name of the “FOLIO-Rancher-Server-nlb”
    a. Provide a password
    b. For the server URL provide https://<DNS name of the “FOLIO-Rancher-Server-nlb-int”>:443
23. You have a rancher server deployed on a single instance.

Deploy AWS EKS cluster from Rancher Server

24. Create VPC with the name “FOLIO-Cluster-VPC”
25. Create a Peering Connection with the name “FOLIO-Cluster-to-Rancher-Server-PC” between VPCs created on the previous steps “FOLIO-Cluster-VPC” & “FOLIO-Rancher-Server-VPC”
26. Create Internet Gateway with the name “FOLIO-Cluster-IGW”
    a. Attach the Internet Gateway to the“FOLIO-Cluster-VPC”
27. Create 6 subnets in 3 different availability zones so that there are 2 subnets (private and public) in each availability zone.
    a. FOLIO-subnet-1
    b. FOLIO-subnet-1-public
    c. FOLIO-subnet-2
    d. FOLIO-subnet-2-public
    e. FOLIO-subnet-3
    f. FOLIO-subnet-3-public
28. Create a route table with the name “FOLIO-Cluster-Public-RT”
    a. Add a route 0.0.0.0/0 -> “FOLIO-Cluster-IGW”
    b. Associate public subnets with the this “FOLIO-fb-Cluster-Public-RT”
        i. FOLIO-subnet-1-public
        ii. FOLIO-subnet-2-public
        iii. FOLIO-subnet-3-public
29. Create a NAT Gateway with the name “FOLIO-Cluster-NAT-GW” for “FOLIO-Cluster-VPC” and “FOLIO-subnet-1-public”
30. Create a route table with the name “FOLIO-Cluster-Private-with-NAT-RT”
    a. Add a route 0.0.0.0/0 -> “FOLIO-Cluster-NAT-GW”
    b. Add a route <FOLIO-Rancher-Server-VPC CIDR> -> “FOLIO-Cluster-to-Rancher-Server-PC”
    c. Associate private subnets with the this “FOLIO-fb-Cluster-Private-with-NAT-RT”
        i. FOLIO-subnet-1
        ii. FOLIO-subnet-2
        iii. FOLIO-subnet-3
31. Open Rancher Web Admin Console in a browser using DNS name of the “FOLIO-Rancher-Server-nlb”
32. Create K8S Cluster using EKS as hosted K8S provider
    a. Press “Add Cluster”
    b. Select “Amazon EKS”
    c. Cluster Name -> FOLIO-Cluster
    d. Select a region where you created “FOLIO-Cluster-VPC”
    e. Provide your Access Key and Secret Key
    f. Press Next
    g. Kubernetes Version -> 1.12
    h. Service Role -> Standard: Rancher generated service role
    i. Press Next
    j. Public IP for Worker Nodes -> No: Private IPs only
    k. VPC & Subnets -> Custom: chose from your existing VPC and Subnets

i. Select “FOLIO-Cluster-VPC”

l. Press Next

m. Subnet ->

i. FOLIO-subnet-1

ii. FOLIO-subnet-2

iii. FOLIO-subnet-3

n. Press Next

o. Security Groups -> default

p. Press Next

q. Configure the desired instance type for Nodes

r. Press Create

33. Wait for Cluster to be provisioned.
34. Open detail for the subnet “FOLIO-subnet-1”
    a. Open Tags tab
    b. Add new tag with the name “kubernetes.io/cluster/folio-cluster” and value “shared”
35. Add the same tag with the value “shared” to other public subnets
    a. FOLIO-subnet-2-public
    b. FOLIO-subnet-3-public
36. You have your k8s cluster deployed

Setup PostgreSQL RDS Server

37. Create a RDS Subnet group with the name “FOLIO-rds-database-SNG”
    a. Select VPC “FOLIO-Cluster-VPC”
    b. Add a private subnet for each AZ
        i. FOLIO-subnet-1
        ii. FOLIO-subnet-2
        iii. FOLIO-subnet-3
38. Create a VPC Security Group with the name “FOLIO-PostgreSQL-DB-SG” linked to “FOLIO-Cluster-VPC”
    a. Add an inbound rule [PostgreSQL, TCP, 5432, “SG created for nodes in an EKS Cluster like folio-cluster-eks-worker-nodes-NodeSecurityGroup...”] - FOLIO DB Access
39. Create a PostgreSQL RDS
    a. DB engine version -> PostgreSQL 11.2-R1+
    b. DB instance identifier -> FOLIO-Database
    c. Master username -> postgres
    d. Master password -> password
    e. Virtual Private Cloud (VPC) -> “FOLIO-Cluster-VPC”
    f. Subnet group -> “FOLIO-rds-database-SNG”
    g. Public accessibility -> No
    h. Availability zone -> No preference
    i. Choose existing VPC security groups -> “FOLIO-PostgreSQL-DB-SG”
    j. Database name -> leave empty
    k. Press “Create database”
40. You have your database created.

Deploy FOLIO platform

41. Clone the git repository folio-install https://github.com/folio-org/folio-install.git
42. Check out branch TODO: branch name
43. Open Rancher Web Admin Console in a browser using DNS name of the “FOLIO-Rancher-Server-nlb”

44. In the top-left main menu select the cluster you created on the previous steps
45. Select “Project/Namespaces”
46. Create a new project “Folio-Project”
    a. Add a new namespace “Folio”
47. In the top-left main menu select the “Folio-Project” you have just created
48. Add Dockerhub and your private Docker registries to the Folio-Project (Resources -> Registries)
49. Select Resources -> Secrets
50. Create a new secret with the name “init-folio-database”
    a. Secret content:
        i. PG_MASTER_USER=postgres
        ii. PG_MASTER_USER_PASSWORD=password
        iii. PG_OKAPI_USER=okapi
        iv. PG_OKAPI_USER_PASSWORD=okapi25
        v. PG_FOLIO_ADMIN_USER=folio_admin
        vi. PG_FOLIO_ADMIN_USER_PASSWORD=password
        vii. PG_DATABASE_OKAPI=okapi
        viii. PG_DATABASE_OKAPI_MODULES=okapi_modules
        ix. PG_DB_HOST=<Endpoint of the PostgreSQL RDS database you created on the previous steps>
51. Build a Docker image “create-database” from the “create-database” folder in the “folio-install”
52. In the Workload of the “Folio-Project” deploy a Job (select the Workload type using “More options” in the top-right corner) using “create-database” docker image with “init-folio-database” secret
53. Create a new secret with the name “okapi-config”
    a. Secret content:
        i. INITDB=true
        ii. PG_HOST=<Endpoint of the PostgreSQL RDS database you created on the previous steps>
        iii. PG_PORT=5432
        iv. PG_USERNAME=okapi
        v. PG_PASSWORD=okapi25
        vi. PG_DATABASE=okapi
        vii. OKAPI_COMMAND=dev
        viii. OKAPI_PORT=9130
        ix. OKAPI_URL=http://okapi:9130
        x. OKAPI_HOST=okapi
        xi. OKAPI_CLUSTERHOST=okapi
        xii. OKAPI_NODENAME=okapi
        xiii. OKAPI_LOGLEVEL=DEBUG
        xiv. OKAPI_STORAGE=postgres
54. Build a Docker image “okapi” from the “okapi” folder in the “folio-install”
55. In the Workload of the “Folio-Project” deploy a Scalable deployment (select the Workload type using “More options” in the top-right corner) with the name “okapi” using “okapi” docker image with “okapi-config” secret
56. Create a new secret with the name “db-connect”
    a. Secret content:
        i. DB_DATABASE=okapi_modules
        ii. DB_HOST = <Endpoint of the PostgreSQL RDS database you created on the previous steps>
        iii. DB_MAXPOOLSIZE = 20
        iv. DB_PASSWORD = password
        v. DB_PORT = 5432

vi. DB_USERNAME = folio_admin
vii. PG_DATABASE = okapi
viii. PG_PASSWORD = okapi25
ix. PG_USER = okapi

57. Clone the git repository platform-complete https://github.com/folio-org/platform-complete
   a. Checkout the branch you need, or just leave master branch
58. Prepare YAML deployment file for back-end modules and deploy back-end modules
   a. Copy files “okapi-install.json” & “install-extras.json” from the platform-complete into the “modules-yaml-generator” folder in the “folio-install”
   b. Open bash terminal and go to the “modules-yaml-generator” folder
   c. Run a docker container using command: **docker run -it --mount src="$(pwd)",target=/usr/local/bin/folio-yaml-builder,type=bind maven:3.6.1-jdk-8-alpine /bin/bash**
   d. Once you have access to the container’s bash run the commands to create the YAML file:
      i. cd /usr/local/bin/folio-yaml-builder/
      ii. mvn clean compile exec:java -Dexec.args="-o=backend-modules.yaml -i=okapi-install.json -i=install-extras.json"
   e. When maven completes the work the resulting file “backend-modules.yaml” will be in your “modules-yaml-generator” folder
   f. Exit the container’s bash, do not forget to remove the container using docker rm <ContainerID>
   g. Open Rancher Web Admin Console in a browser using DNS name of the “FOLIO-Rancher-Server-nlb”
   h. In the Workload of the “Folio-Project” press “Import YAML” button and paste the content of the “backend-modules.yaml” or use “Read from a file” button
      i. Import mode -> Namespace: Import all resources into a specific namespace
      ii. Default Namespace -> folio
      iii. Press “Import” button
   i. Wait until all containers are created. It can take a while.
59. Create a new secret with the name “diku-tenant-config”
   a. Secret content:
      i. ADMIN_PASSWORD=admin
      ii. ADMIN_USER=diku_admin
      iii. OKAPI_URL=http://okapi:9130
      iv. PURGE_DATA=true
      v. REF_DATA=true
      vi. REGISTRY_URL=http://okapi:9130/_/proxy/modules
      vii. SAMPLE_DATA=true
      viii. TENANT_DESC=Danish Library Technology Institute
      ix. TENANT_ID=diku
      x. TENANT_NAME=Datalogisk Institut
60. Build a Docker image “create-tenant” from the “create-tenant” folder in the “folio-install”
61. In the Workload of the “Folio-Project” deploy a Job (select the Workload type using “More options” in the top-right corner) using “create-tenant” docker image with “diku-tenant-config” secret
62. Build a Docker image “create-deploy” from the “create-deploy” folder in the “folio-install”
   a. Beforehand copy files “okapi-install.json”, “install-extras.json”, “stripes-install.json” from the platform-complete into the “create-deploy/install” folder in the “folio-install”
63. In the Workload of the “Folio-Project” deploy a Job (select the Workload type using “More options” in the top-right corner) using “create-deploy” docker image with “diku-tenant-config” secret
   a. It will take some time (for me it was up to 3 hours) to complete the job

64. Build a Docker image “bootstrap-superuser” from the “bootstrap-superuser” folder in the “folio-install”
65. In the Workload of the “Folio-Project” deploy a Job (select the Workload type using “More options” in the top-right corner) using “bootstrap-superuser” docker image
66. Create a load balancer for OKAPI
    a. In the Workload of the “Folio-Project” go to “Service Discovery”
    b. Press “Add Record” button
    c. Fill data for a new record
        i. Name -> okapi-nlb
        ii. Namespace -> folio
        iii. Resolves to -> One or more workloads
        iv. Press “Add Target Workload”
        v. Select a Workload -> okapi
        vi. Press “Show advanced options”
        vii. As a -> Layer-4 Load Balancer
        viii. Press “Add Port” button under “Port Mapping”
        ix. Port Name -> port-9130
        x. Publish the service port -> 9130
        xi. Protocol -> TCP
        xii. Target Port -> leave “Save as service port”
        xiii. Node Port -> leave “Random”
        xiv. Expand “Labels & Annotations”
        xv. Press “Add Annotation” button
        xvi. Key -> service.beta.kubernetes.io/aws-load-balancer-type
        xvii. Value -> nlb
        xviii. Press “Create” button
67. Check a Network Load Balancer for okapi on the AWS console
    a. Go to EC2 services -> Load Balancing -> Load Balancers
    b. Check the newly created load balancer with an ugly name, wait until it becomes active
    c. Copy the DNS name and run CURL
        i. curl -w '\n' http://<your_DNS_name>:9130/_/proxy/modules
68. Build a Docker image “stripes-diku” from the “stripes-diku” folder in the “folio-install”
    a. Beforehand copy files from the platform-complete into the “stripes-diku” folder in the “folio-install”
        i. *.json -> “stripes-diku” folder
        ii. *.js -> “stripes-diku” folder
        iii. Yarn.lock -> “stripes-diku” folder
        iv. tenant-assets/* -> “stripes-diku/tenant-assets” folder
    b. Beforehand change the Dockefile and setup a correct value for OKAPI_URL argument using DNS name of the load balancer created for okapi
        i. For example: **ARG OKAPI_URL=http://ad2232214821b11e99fc006191d465ba-a91a08bd720aef8d.elb.us-west-2.amazonaws.com:9130**
    c. Build a docker image
69. In the Workload of the “Folio-Project” deploy a Scalable deployment (select the Workload type using “More options” in the top-right corner) with the name “stripes-app” using “stripes-diku” docker image
70. Create the L-4 load balancer for “stripes-app” the same way as you have already created one for okapi, just use port 80 instead of 9130
71. Wait until the newly create load balancer becomes active
72. Check the DNS name of the load balancer and open Stripes UI in the browser using.

73. You have deployed FOLIO Platform!